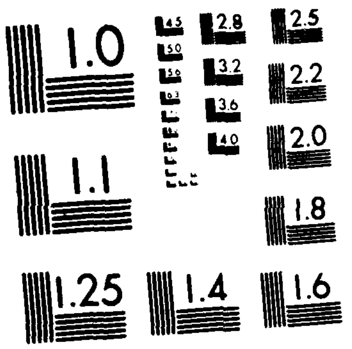
AD-A191 587 SLAPP: A SPECIAL MULTIPROCESSOR ARRAY FOR SIGNAL PROCESSING AND LINEAR ALGEBRA(U) J J SYMANSKI ET AL. DEC 87 1/1

UNCLASSIFIED F/G 12/6 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

AD-A191 587

PORT DOCUMENTATION PAGE

SCHEDULE	1b. RESTRICTIVE MARKINGS
	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center — 6b. OFFICE SYMBOL (if applicable) NOSC	7a. NAME OF MONITORING ORGANIZATION Naval Ocean Systems Center
6c. ADDRESS (City, State and ZIP Code) San Diego, California 92152-5000	7b. ADDRESS (City, State and ZIP Code) San Diego, California 92152-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research — 8b. OFFICE SYMBOL (if applicable) ONR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) 800 N. Quincy Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	AGENCY ACCESSION NO.
61153N	EE55	RR01509	DN088 669

11. TITLE (include Security Classification)
SLAPP: A special multiprocessor array for signal processing and linear algebra

12. PERSONAL AUTHOR(S)
J.J. Symanski, T. Henderson, J. Shirasago, J. Celto, and B. Drake

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Professional paper/speech	FROM Aug 1987 TO Aug 1987	December 1987	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
systolic array, RAM, algorithms, boundary processors

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The key to meeting the extremely high throughput requirements of future military signal and image processing systems is parallelism in algorithms and hardware. This paper will describe the implementation of a core set of algorithms on one possible hardware implementation, designed to achieve high speed and efficient parallelism. This approach and design procedure, while using currently available integrated circuit building blocks, is similar to how this type of processor will be developed in the future using VLSI.

[illegible]

Presented at SPIE International Technical Symposium, 17-21 Aug 1987, San Diego, CA.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL J.J. Symanski	22b. TELEPHONE (include Area Code) 619-553-2530 — 22c. OFFICE SYMBOL Code 741

To appear in the Proceedings of the SPIE International Technical Symposium, Vol. 826-32, Advanced Algorithms and Architectures for Signal Processing II, San Diego, CA 18-19 August 1987.

826 32

SLAPP: A special purpose multiprocessor array for signal processing and linear algebra

J. J. Symanski, Tom Henderson, Judy Shirasago, John Celto, Barry Drake

Naval Ocean Systems Center
San Diego, CA 92152

ABSTRACT

The key to meeting the extremely high throughput requirements of future military signal and image processing systems is parallelism in algorithms and hardware. This paper will describe the implementation of a core set of algorithms on one possible hardware implementation, designed to achieve high speed and efficient parallelism. This approach and design procedure, while using currently available integrated circuit building blocks, is similar to how this type of processor will be developed in the future using VLSI.

1. INTRODUCTION

It is now almost a decade since Kung and Leiserson[1] first described a systolic array. Many advances have been made and several machines are in use[2]. Algorithms are under development which will take advantage of the parallelism available in systolic and other architectures. This paper will describe work aimed specifically at efficient implementation of a core set of algorithms for use in linear algebra and signal processing. The architecture of the SLAPP has been described[3] previously. This paper will describe data movement in the algorithms of interest, namely, matrix multiplication, QRD, SVD and generalized SVD, as well as some details of a bit-slice microprogrammed implementation of the SLAPP.

2. MATRIX MULTIPLICATION

Matrix multiplication is the easiest algorithm to implement due to its regularity and use of only multiplies and adds. It requires n^3 operations which can be done in O(n) time on n^2 processors. There are several ways to implement the algorithm which depend on whether the data is outside the array or has been loaded into the array. For data coming from outside the array the engagement process of Speiser and Whitehouse[4] requires 3n-2 steps. For data within the array, the process described by Symanski[5] requires 2n-1 steps. The SLAPP array could implement the engagement algorithm more easily by not requiring skewing of the data by the host which complicates the host IO and programming. The data could simply be dumped into the top row and left column of the SLAPP array and stored in auxiliary or dual port RAM. Appropriate routines and control message passing could distribute the data down the columns and across the rows to complete the algorithm.

3. QR DECOMPOSITION (QRD)

The QRD algorithm of F. T. Luk[6] utilizes two types of processors: boundary processors which lie on the main diagonal and internal processors. Sine/cosine pairs for each row are generated in the boundary processors and then passed down the row to be applied to the internal processors. Each processor operates on a 2x2 sub-matrix. Data is input to the systolic array through the top processors. The data is time skewed, so that each pair of columns is skewed one row. This skewing ensures that the corect sine/cosine pair operates on the appropriate two rows of the data matrix.

A C program was written to simulate the data movement of the above QRD algorithm using the unified architecture. For an M x 8 data matrix with the unified architecture, there are two virtual processors per physical processor. Each virtual processor contains seven elements: the 2x2 sub-matrix [UL, UR, LL, LR], sine, cosine, and temp. Temp is a memory buffer that exists for correct data movement. The lower right (LR) element of the 2x2 sub-matrix must be held in each processor one additional time step so that the correct data gets moved to the next processor at the appropriate time. Only the data in the LL and LR elements move. The UL and UR elements are updated until they converge to R, the upper triangular result. Order of movement for the LL and LR elements are as follows:

$$\begin{aligned} &LL \rightarrow LR \\ &TEMP \rightarrow LL \\ &LR \rightarrow TEMP \end{aligned}$$

Since there are two virtual processors per physical processor, there is data movement within as well as between physical processors. After the sine/cosine pairs are generated and applied to the boundary processors, they are passed one processor to the right every time step to be applied to the internal processors. See Figures 1 and 2.

826 32

A C simulation is being written for the data movement of the SVD algorithm using the unified architecture. For an M x 8 upper triangular matrix, there are three virtual processors [two even and one odd] in the physical processors on the main diagonal and two virtual processors [one even and one odd] in the internal physical processors. Each virtual processor contains eight elements: the 2x2 submatrix [UL, UR, LL, LR] and the two sine/cosine pairs [$\cos_j$, $\sin_j$, $\cos_k$, $\sin_k$]. Data movement is between as well as within physical processors. $\cos_j/\sin_j$ are passed horizontally one virtual processor to the right and $\cos_k/\sin_k$ are passed vertically one virtual processor upward. See Figures 3 and 4.

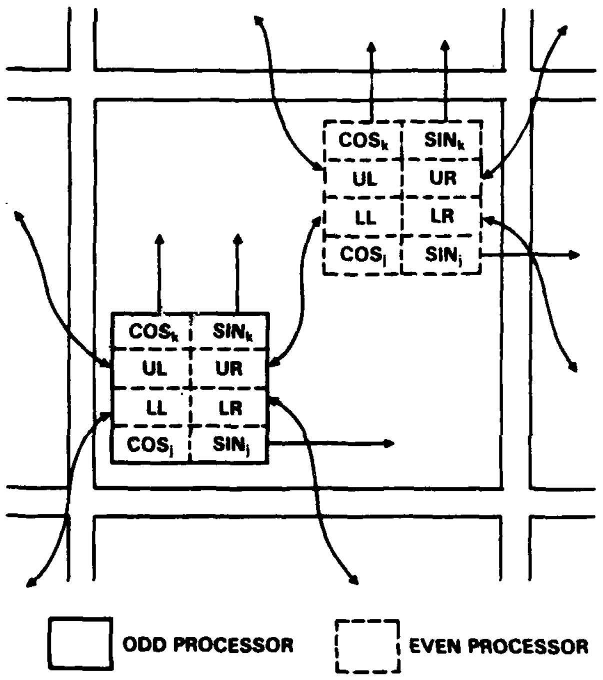


Figure 3. SVD Data Flow in the SLAPP Processor

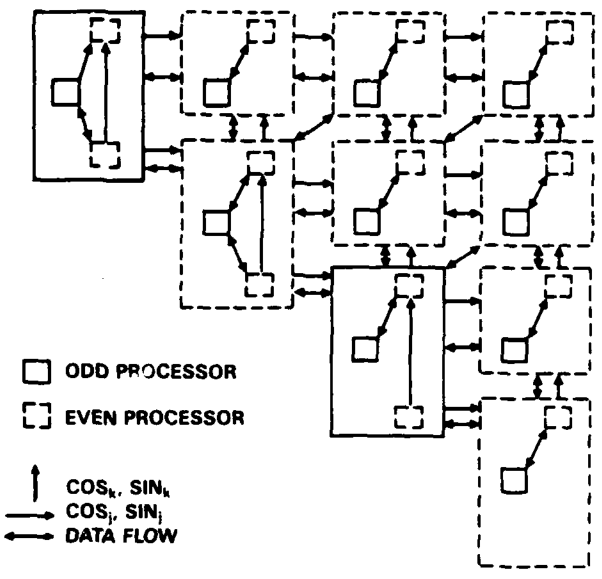


Figure 4. SVD Data Flow in the SLAPP Array

As mentioned above, there are two sets of sine/cosine rotations generated. Equations used in the simulation for the sine cosine generations are:

Luk's Equations:

$\text{srho} = (w + z) \div x;$
$\sin_s = \text{sign(srho)} \div \text{sqrt}(1 + \text{srho}^2);$
$\cos_s = \sin_s \bullet \text{srho};$

$\text{den} = 2 \bullet \sin_s \bullet w;$
$\text{num} = \cos_s \bullet (z - w) + \sin_s \bullet x;$
$\text{krho} = \text{num} \div \text{den};$
$t = -\text{sign(krho)} \bullet [\text{abs(krho)} + \text{sqrt}(1 + \text{krho}^2)];$
$\cos_k = 1 \div \text{sqrt}(1 + t^2);$
$\sin_k = \cos_k \bullet t;$

Overflow Equations:

$\text{srho} = x \div (w + z);$ $[\text{if } |x| < |w + z|]$
$\cos_s = 1 \div \text{sqrt}(1 + \text{srho}^2);$
$\sin_s = \cos_s \bullet \text{srho};$

$\text{den} = 2 \bullet \sin_s \bullet w;$ $[\text{if } |\text{den}| < |\text{num}|]$
$\text{num} = \cos_s \bullet (z - w) + \sin_s \bullet x;$
$\text{krho} = \text{den} \div \text{num};$
$t = [-\text{sign(krho)} \bullet \text{abs(krho)}] \div (1 + \text{sqrt}(1 - \text{krho}^2));$
$\sin_k = -\text{sign(krho)} \div \text{sqrt}(1 + t^2);$
$\cos_k = \sin_k \bullet t;$

826 32

$\cos_j = \cos_s \bullet \cos_k - \sin_s \bullet \sin_k;$
$\sin_j = \cos_s \bullet \sin_k + \sin_s \bullet \cos_k;$

$$K = \begin{bmatrix} \cos_k & \sin_k \\ -\sin_k & \cos_k \end{bmatrix} \qquad S = \begin{bmatrix} \cos_s & \sin_s \\ -\sin_s & \cos_s \end{bmatrix} \qquad J' = \begin{bmatrix} \cos_j & -\sin_j \\ \sin_j & \cos_j \end{bmatrix}$$

$$A = \begin{bmatrix} w & x \\ y & z \end{bmatrix}$$

Diagonalization of a 2x2 matrix is done in two steps: symmetrization and annihilation [$D = J' \bullet A \bullet K$]. K is the annihilation rotations matrix and S is the symmetrization rotations matrix. In the simulation, the sines and cosines for J are computed from the sines and cosines of S and K. This saves one matrix-matrix multiply.

For matrices with a column dimension higher than the number of columns in the systolic array, the physical processors will have more virtual processors in them as in the QR. For an M x 16 upper triangular matrix, the physical processors on the main diagonal would now have eight virtual processors [five even and three odd] in them while the internal physical processors would contain four each of the odd and even processors.

GENERALIZED SVD(GSVD)

Two preparatory QRD's are required for the computation of the GSVD. One matrix is fed into each of the arrays: A into array Ta and B into array Tb. After the QRD of each matrix is computed, an implicit SVD of $R_a R_b^{-1}$ begins. For simplicity we consider only one odd rotation of a typical sweep. Also, R_a, R_b are 2 x 2 submatrices of R_A, R_B, respectively.

After the elements in the boundary processors of, say, array Tb have been communicated to the boundary processors of array Ta, a 2x2 matrix C is formed in each of the overlapped boundary processors. The elements of C in a typical boundary processor are given by

$$c_{kl} = \sum_{i}^{i+1} ra_{ki}\, adj(R_b)_{il}$$

where the adjugate matrix adj (X) = $Det(X)\, X^{-1}$ is computed to avoid computation of the determinant. C is upper triangular since the *submatrices R_a and R_b are upper triangular in the boundary processors*. Next, rotation sines and cosines are computed to zero out the off diagonal elements of C in each boundary processor as described in [7,9]. The equation

$$\begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} = J^t C K$$
$$= J^t R_a R_b^{-1} K$$
$$J^t R_a (K^t R_b)^{-1}$$

shows where to send these rotation sines and cosines within the arrays Ta and Tb. The c_j, s_j pair propagate to the east through array Ta where they are applied to the elements of R_a. Likewise, the c_k, s_k pair is sent east in array Tb where they are applied to the elements of R_b.

Since the previous rotations destroy the upper triangular structure of $J^t R_a$ and $K^t R_b$, a rotation matrix Q is computed to return both to upper triangular form. This is possible because the rows of R_a and R_b are parallel. Thus,

$(J^t R_a)Q$, $(K^t R_b)Q$

are upper triangular, where Q can be computed in the overlapped boundary processors either from R_a or R_b. c_Q, s_Q are sent north through both Ta and Tb where they are applied to R_A and R_B, respectively. The algorithm continues in this way through the odd-even iterations until R_A and R_B have parallel rows, i.e.,

$U^t(R_A)((R_B)^{-1})V = D$

$U^t(R_A) = DV^t(R_B)$

where D is diagonal and, hence the rows of the LHS are just a scalar multiple of the rows of the RHS.

SLAPP UNIFIED ARCHITECTURE

The SLAPP bitriangular architecture has been described previously[1]. Now we will describe the unification of the matrix-multiplication, QRD, SVD and GSVD onto the SLAPP architecture.

826 32

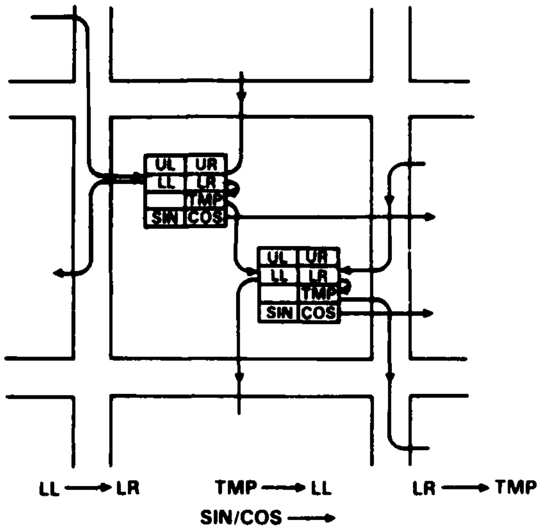


Figure 1. QRD Data Flow in the SLAPP Processor

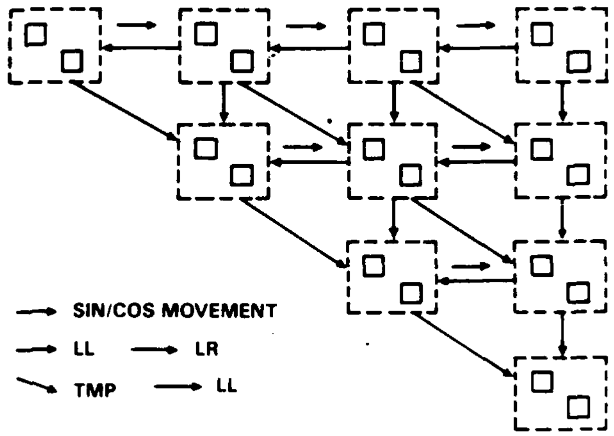


Figure 2. QRD Data Flow in the SLAPP Array

The equations used in the simulation for the sine cosine generation are:

Luk's Equations:

rho = w − y;
sin = sign(rho) − sqrt(1 + rho^2);
cos = rho • sin;

Overflow Equations: (if y < eps • |w|)

rho = y − w;
cos = 1 − sqrt(1 + rho^2);
sin rho • cos;

where w, x, y, and z are the elements of the 2x2 sub-matrix.

$$\begin{bmatrix} w & x \\ y & z \end{bmatrix}$$

For matrices with a column dimension higher than the number of columns in the systolic array, each physical processor would contain more virtual processors. A simulation was also written for the data movement of an M x 16 data matrix using the unified architecture. There are now eight virtual processors per physical processor instead of two. For an M x 32 data matrix, there would be 32 virtual processors per physical processor. One major difference in these simulations is there are now internal as well as boundary virtual processors in the physical boundary processors.

4. SINGULAR VALUE DECOMPOSITION (SVD)

The SVD algorithm of F. T. Luk[6,7] is an algorithm for an n x n upper triangular matrix R. It is based on the odd-even ordering of Stewart[8]. Like the QR algorithm above, sine/cosine pairs are generated to the boundary processors. These rotations are then applied to the rows and columns associated with that processor. Again, each processor operates on a 2x2 submatrix.

With the odd-even ordering, there are two kinds of processors: odd processors are those with the upper left element of the 2x2 submatrix being odd and even processors are those with an even upper left element. The algorithm begins with an odd rotation followed by an even rotation. This pattern continues until convergence. A rotation consists of the following: sine cosine pairs are computed in the odd (even) boundary processors and applied. Data in the boundary processors is sent to even (odd) processors to the northwest, northeast, and southeast and the sine/cosine pairs are passed to neighboring odd (even) processors to the north and east. After being applied to these internal processors, data is sent to even (odd) processors to the northwest, northeast, southwest, and southeast and the sine cosine pairs are passed to the north and east. The even (odd) rotation can now begin since the boundary processors have the necessary data to start computing their sine/cosine pairs. Note that the sine/cosine pairs in the odd (even) processors are still being propagated up and to the right to the rest of the odd (even) internal processors. This occurs since a processor can begin working as soon as it receives all data and rotations from its neighboring processors.

826 32

Speiser and Whitehouse present the engagement array[4] algorithm for multiplying two matrices in time 3n − 2, where n is the dimension of the matrix. The input matrices, say A and B, are skewed such that the successive columns of A and rows of B are delayed one time step at successive processors. Partial products are accumulated in each processor as the matrices flow through the array. At completion the result matrix is stored in the systolic array. For many matrix-matrix multiplies, such as those required for the MUSIC high resolution direction finding algorithm[10], the matrices could be 'stacked' with results offloaded as each matrix-matrix multiply is completed.

The Luk ORD and SVD algorithms require two, slightly different, systolic array architectures (Figures 5a and 5b). As more processors are added, to accommodate larger matrices, more "SVD processors" are required than "QRD processors" to solve the same size problem. This results in some "SVD processor" idle time while the QRD is proceeding. Also, due to odd-even ordering for the SVD algorithm, half of the processors are idle during odd (even) cycles. The unified architecture is a mapping of the Luk QRD and SVD architectures onto a Gentleman-Kung triangular array, with the advantage of greater hardware efficiency using fewer processors.

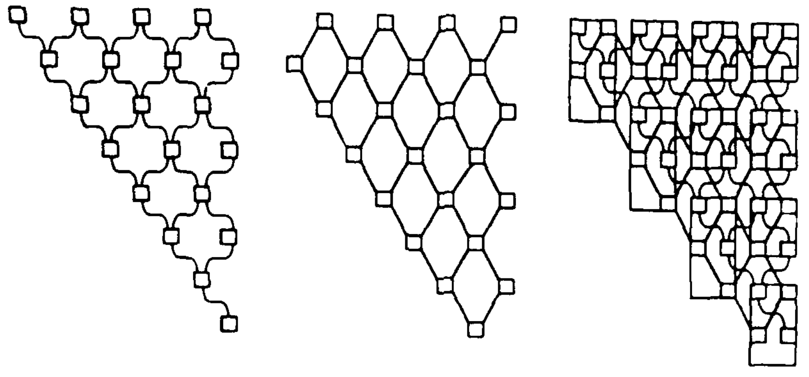


Figure 5. The Unified Architecture

For an m x 8 problem, the QRD array is composed of 20 processors while 23 processors are required for the SVD. Superimposing the two arrays and grouping together "SVD processors" with "QRD processors" results in the unified architecture (Fig. 5c). Now the Luk "QRD and SVD processors" are virtual cells of a unified processor. The number of processors required to factor an m x 8 matrix is reduced from 23 to 10.

With point-to-point connections between the two triangular arrays the bitriangular architecture is configured to compute the GSVD in a straight forward way. Point-to-point connections arise from folding the two triangular arrays together and connecting corresponding processors, giving a three dimensional architecture as depicted in Fig. 6.

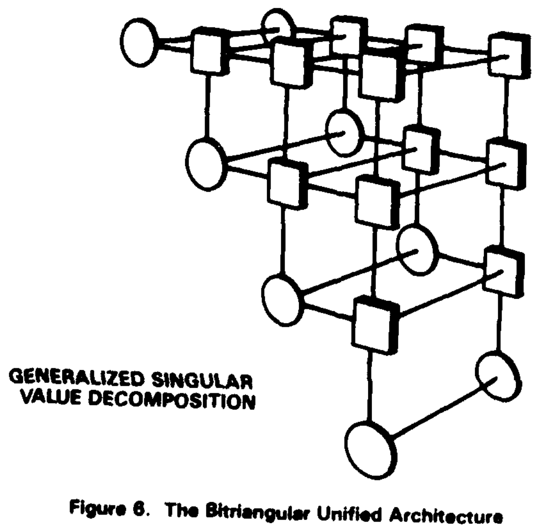


Figure 6. The Bitriangular Unified Architecture

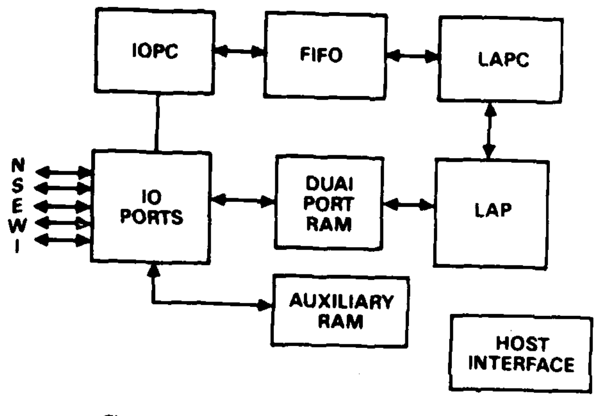


Figure 7. The SLAPP Node Architecture

826 32

SLAPP NODE ARCHITECTURE

The architecture of the SLAPP node is shown in Figure 7. The components contained in each block will be explained below.

The Input/Output Processor Control (IOPC) consists of a sixteen bit sequencer (IDT49C410), test circuitry for determining the state of the IOP and determining conditions for microprogram jumps. Also in this block is the control microcode and other control circuitry.

The five IO Ports with appropriate buffer and control circuitry is contained in the IO Ports block. This circuitry is mostly SSI buffers and control circuitry for fast block IO transfer control. This accounts for a large number of packages in a SSI design. For highest speed a full 32 bit parallel approach is best but requires many pins. Pin count could be brought down significantly by four bit wide data paths or bit serial IO. But IO transfer rate suffers. The IO transfer rate may not be the limiting factor in some applications so the narrowing of the IO path width would be a good trade-off parameter.

The FIFO block contains two FIFOs, one for communication of commands from the IOPC to the Linear Algebra Processor Controller (LAPC). The IOPC will tell the LAPC that data has arrived for a particular operation and the LAPC will execute the appropriate code. When the LAPC has finished the computations, it sends a status word to the IOPC via the second FIFO to inform the IOPC that data can be moved out or a new operation can be started.

The LAPC is similar to the IOPC in that it is a 16 bit microsequencer with testing capability to determine what needs to be done and control circuitry and to perform the necessary actions.

The Dual Port RAM acts as the data storage and serves to decouple the two processors from one another and simplify programming and data fetching. The dual port RAM used in the simulation is the IDT7132/42 16K (2kx8). There are two 2kx32 bit sections of dual port RAM, each connected to a separate bus. This is to allow I/O with two of the five ports simultaneously as well as to supply two operands to the arithmetic processor in one cycle.

The Linear Algebra Processor (LAP) is the computation unit for the SLAPP node. It consists of the Bipolar Integrated Technology (BIT) B2110/2120 multiplier and ALU, plus a small register file for temporary storage during computations. The multiplier and accumulator chips together contain about 125,000 devices.

The Auxiliary RAM is made up of high density static RAM for storage of large amounts of data which may come into the node during computations and to allow distributed memory to minimize array IO. Control circuitry for DMA transfers could take 20 IC's or a PLD.

A rough count of the gates and device equivalents is shown in Table 1. The device estimates use four devices per gate and one device per RAM memory bit. Note that the multiplier and accumulator chips account for almost three quarters of the gate count. But when memory is included, the microcode and RAM require about eight times as many devices as the logic. The relative high density of memory to the lower density of gates and wiring may make this comparison less important since silicon area is the important issue rather than device count.

Table 1. The SLAPP Processor Gate and Device Count

BLOCK	ICs(VLSI)	GATES	RAM
IOPC	36 (6)	2,800	384 Kbits
FIFO	10	–	1 Kbits
LAPC	30 (10)	2,600	640 Kbits
IO PORTS	30	2,400	–
DUAL PORT RAM	30 (8)	1,000	128 Kbits
AUXILIARY RAM	15 (4)	500	256 Kbits
LAP	16 (2)	33,000	–
HOST INTERFACE	10 (2)	500	128 Kbits
	177 (32)	42,800 gates	1,537,000 bits
		171,200 devices	1,537,000 devices

Total devices: 1,700,000

Calculations use 4 transistors per gate and 1 transistor per RAM bit.

Also, the divide and square root circuitry within the multiplier chip requires only about 14,000 devices out of the 125,000 devices in the floating point chip set[11], about 12% of the total logic device count and less than 1% of the total device count of the SLAPP node. This being the case, the use of less capable internal processors which only multiply and add, does not seem warranted. This is especially true if one wishes to reconfigure the array in some manner due to application requirements or node failure. Having all nodes be boundary processors simplifies system design and yields far better system flexibility and fault tolerance.

826 32

6. IMPLEMENTATION

Due to the high speed requirements, complexity and longevity of military systems, the NOSC work has emphasized a high speed, programmable processor. When working at the early stages of algorithm development, there is really no way to specify exact speed requirements. Real systems are far too complex and involve too many operations to give exact estimates as to the speed requirements of a particular algorithm. In general, there is never enough speed or memory. If one is dealing with an application which is very well understood and the hardware and software will never be changed, it may be possible. This is rarely the case in complex military systems.

The physical size of the system can be critical in some applications. Fabricating an array with commercially available devices, the SLAPP processor would require about 200 integrated circuit packages. The same design could be reduced to about 40 packages using gate arrays with five to ten thousand gates each. The current processor design utilizes 32 LSI devices. The gate arrays would replace only the simpler components.

The processor could be implemented in one package if full custom or a megacell technology were used. Rough estimates of the gate and memory requirements indicate that the processor would require less than a quarter of a square inch of silicon using 0.5 micron CMOS technology proposed for VHSIC Phase 2 in the 1990s[12].

Design time grows as the size shrinks. Even with the availability of powerful CAE tools, the complexity of the systems tends to increase errors and prolong completion. If the correct tools are not available at the beginning of the development, the acquisition of the tools and training in the efficient use of these complex tools, takes significant time.

Programming of sophisticated routines can be difficult, especially at the microcode level. For efficient implementation, it usually must be done for each macro routine by a programmer experienced in the hardware resources of the machine. A powerful compiler could be written to generate microcode using a high level language, but the writing of an efficient compiler is no simple task in itself. In designing a processor to efficiently implement routines it is highly desirable to actually test the routines on a simulation of the hardware to avoid costly changes and iterations later in the hardware. The addition of a register file or another data path or more memory can be costly once the fabrication process has begun. Program verification can be done with the appropriate CAE tools. Of course, if a large user community is to be supported, a high level language is essential. The Warp and iWarp work at Carnegie Mellon is an excellent example of this kind of large scale development.

Simple tools for writing microcode for the SLAPP node were written in C in about a man-month. There are four programs. One for generating microcode for the IOP and a second for generating code for the LAP. These two programs take text files written with an ordinary ASCII editor and convert the text to binary files used by the simulation program. A third program converts floating point numbers into binary data files that the logic simulation program can use for input and a fourth program converts the binary data files generated by the simulation to IEEE floating point numbers the user can read easily. These tools make generating and interpreting simulation data much easier.

7. SUMMARY

A core set of algorithms useful for many applications in signal processing has been developed and mapped onto a programmable high speed parallel processing architecture. Simulations of the algorithms have been completed on the IBM PC in PC Matlab and in the C language. Unique features of the SLAPP array are 1) independent IO and computation sequencers in each processor to achieve highest system efficiency and ease programming, 2) a high speed, non-pipelined arithmetic unit capable of performing divides and square roots in hardware. A gate level simulation of a dual microsequencer implementation of the architecture has been entered into a CAE workstation and logic simulation of processor operation has started.

8. ACKNOWLEDGMENTS

This work is funded by Dr. Richard L. Lau of the Office of Naval Research, Code 1111.

The concepts presented are the result of discussions with many people. The authors would like to thank co-worker at NOSC Jeffrey Speiser for help in understanding the algorithms. Special thanks to Professor Frank Luk and researchers at Cornell for informative discussions about the algorithms. The author also gratefully acknowledges many stimulating discussions with Professor H. T. Kung and members of the WARP project at Carnegie Mellon University.

826 32

9. REFERENCES

1. H. T. Kung and C. E. Leiserson, "Systolic Arrays for VLSI", in I. S. Duff and G. W. Stewart, Sparse Matrix Proceedings, 1978, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979. (Reprinted in Mead and Conway, Introduction to VLSI, Addison and Wesley, 1980).

2. Computer, July 1987, published by the Computer Society of the IEEE, Vol. 20, No. 7.

3. Symanski, J. J., "Architecture of the Systolic Linear Algebra Parallel Processor (SLAPP)", Proceedings of the SPIE International Technical Symposium, Vol. 698-34, Real Time Signal Processing, San Diego, CA 17-22 August 1986.

4. Speiser, J. M. and H. J. Whitehouse, "Parallel Processing Algorithms and Architectures for Real Time Signal Processing", Proceedings of the SPIE International Technical Symposium, Vol. 298-01, Real Time Signal Processing IV, San Diego, CA 25-28 August 1981.

5. Symanski, J. J, "Implementation of Matrix Operations on the Two-Dimensional Systolic Array Testbed", Proceedings of the SPIE International Technical Symposium, Vol. 431-19, Real Time Signal Processing, San Diego, CA 21-26 August 1983.

6. Luk, F. T., "A Triangular Processor Array for Computing the Singular Value Decomposition", Linear Algebra and Its Applications, Vol. 77, May 1986, pp. 259-273.

7. Luk, F. T., "Architectures for Computing Eigenvalues and SVDs", Proc. SPIE, Vol. 614, pp. 24-33, August 1986.

8. Stewart, G. W., "A Jacobi-like Algorithm for Computing the Schur Decomposition of a Non-Hermitian Matrix", SIAM J. Sci. Statist. Comput., Vol. 6, No. 4, October 1985, pp. 853-864.

9. Drake, B. L., F. T. Luk, J. M. Speiser and J. J. Symanski, "SLAPP: A Systolic Linear Algebra Parallel Processor", IEEE Computer, Vol. 20, No. 7, pp. 45-49, July 1987.

10. Schmidt, R. O., "Multiple Emitter Location and Signal Parameter Estimation", IEEE Trans. Antennas and Propagation, Vol. AP-34, No. 3, pp 276-280, March 1986.

11. B. Elkind, J. Lessert, J. Peterson, G. Taylor, " A Sub 10 nS Bipolar 64 Bit Integer/Floating Point Processor Implemented on Two Circuits", presented at the IEEE Bipolar Circuits and Technology Meeting, Minneapolis, Minn., 21-22 Sept 1987.

12. "TRW's Superchip Passes first Milestone", Electronics, 10 July 1986.